Autoformalization using Formal Proof Sketches

Freek Wiedijk

Radboud University Nijmegen
Institute for Computing and Information Sciences

Abstract

We present some experiments in autoformalization in which an intermediate stage of formal proof sketches forces the AI to keep a tight link between the source text and the formalization. One of the key notions of our approach is not to have several files, but to have the AI produce a single 'master' file from which everything is derived: a LaTeX version of the original, a 'blueprint' version in which all details have been filled in, a version with formal proof sketches, and the final formalization.

The aim of our project is to get the AI to be able to formalize a mathematical text autonomously, without any human direction or feedback, starting from just a scan of the original, and resulting in a correct and complete formalization of the full text.

When we write 'the AI' in the abstract above, we mean a state-of-the-art LLM. Currently we use Claude Opus [5], sponsored by Josef Urban from his ERC grant NextReason.

The key notion of our presentation is that of a 'formal proof sketch' [3]. This is closely related to a 'sorried' version of an unfinished formalization, but there is also an important difference (see the first point below). A formal proof sketch presupposes a proof language that supports a declarative style and that is close to a controlled natural language. Examples of such proof languages are those of the Mizar system and the Isar language of the Isabelle system.

There are two sides to a formal proof sketch:

- They are unfinished formalization proofs, but with the text *as close to the original English text* as possible. Every step in the original proof becomes a step in the formal proof sketch, and every statement in the original proof is formalized as closely as possible by a formal statement. If done correctly, a formal proof sketch really reads like the original.

- They are *formal*. Although the correctness of a formal proof sketch is not decidable in general, it does have a notion of formal correctness. This is defined as that it is possible to obtain an axiom- and sorry-free formalization by replacing all 'sorries' with subproofs as well as inserting intermediate steps between the formal proof sketch steps. That there is a correctness notion is important: these are not just proof sketches, they are *formal* proof sketches.

The pipeline in our approach is to have the AI do all of the following transformations:

scanned pages of a mathematical text
↓
LaTeX reproducing the scanned pages
↓
a 'blueprint' = a version of the text with all details added
↓
an unfinished formalization in which the 'proofs' are formal proof sketches
↓
full formalization, correct without any errors or warnings

The essential property of this pipeline is that each of these steps is *small*. We require the LaTeX to reproduce the scan with all the page breaks and even the line breaks intact. We require the blueprint to fully contain the LaTeX version of the text as a subset. We require the formal proof sketches to match the English text version of the proofs as closely as possible (cf. the first point above). And in the final formalization we require the proofs to have the formal proof sketch version unmodified as a skeleton.

In our implementation, the LaTeX, blueprint, formal proof sketch and final formalization versions are not kept as separate files, but as views of a single 'master' file. Post-edit hooks ensure that the master file stays well-structured and that the views remain consistent with it.

Within the pipeline the AI is allowed to iterate as much as it needs to. The small-step structure is what makes that iteration self-contained, so that no human direction is required. It forces a tight link between the source text and the formalization at every stage. This tightness both increases accuracy and keeps the AI-produced output human-accessible. Having the formal proof sketch as a distinct intermediate also matters because the AI handles original-matching and detail-filling differently: some transformations are mostly about matching the original text faithfully, and some about filling in details correctly. Having separate artifacts in between keeps the two skills apart, and that the last two are system-checkable helps even more.

The pipeline can also be run in reverse. In the long run, when AI begins producing mathematics that humans have a hard time following, this could be a way to 'explain' it back into texts accessible to humans.

This work is closely related to the work by Jiang et al. [4]. Our focus though is that the formal proof sketch version of the formalization is the central artifact, not just an intermediary. If a user wants to check what the AI has produced, they will look at this version, and not at the checkable final version. Another difference from Jiang et al. is the master-file architecture as a way to force the AI not to wander off the proper path.

At the time of writing we only have some preliminary experiments. We are planning a bigger experiment formalizing Newman's classical book on plane topology [1] using formal proof sketches written in Isabelle's Isar language [2].

References

[1] M. H. A. Newman. *Elements of the Topology of Plane Sets of Points*. Cambridge University Press, 1939. Second edition 1951, reprinted 1961.

[2] M. Wenzel. Isar — A Generic Interpretative Approach to Readable Formal Proof Documents. In: Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin-Mohring, L. Théry (eds.), *Theorem Proving in Higher Order Logics (TPHOLs '99)*, Lecture Notes in Computer Science 1690, pp. 167–183, Springer, 1999.

[3] F. Wiedijk. Formal Proof Sketches. In: S. Berardi, M. Coppo, F. Damiani (eds.), *Types for Proofs and Programs (TYPES 2003)*, Lecture Notes in Computer Science 3085, pp. 378–393, Springer, 2004.

[4] A. Q. Jiang, S. Welleck, J. P. Zhou, T. Lacroix, J. Liu, W. Li, M. Jamnik, G. Lample, Y. Wu. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs. In: *International Conference on Learning Representations (ICLR)*, 2023.

[5] Anthropic. Introducing Claude 4. https://www.anthropic.com/news/claude-4, May 2025.